

A Fast SVM-based Feature Selection Method, Combining MFE (Margin-Maximizing Feature Elimination) and Upper Bound on Misclassification Risk

Yaman Aksu

October 17, 2012

Abstract

We introduce a fast and computationally low-cost SVM-based novel feature elimination method, and show it is effective on high-dimensional data with e.g. thousands of features. The method’s premise is to utilize light classifier retraining, rather than full SVM retraining, at each feature elimination step and be consistent with both margin maximization central to SVM learning and a well-known upper bound in Statistical Learning Theory on the expected risk of making a classification error. On high-d gene and other datasets, we show that our proposed method achieves higher or competitive generalization accuracy, lower or similar test set classification error rate, than earlier MFE-based methods and RFE, even when the earlier methods stepwise utilize full SVM retraining i.e. re-learn SVM weights from scratch. We find that our proposed method and the previously proposed MFE-LO are the two best performers with respect to generalization accuracy, and point out some limitations of MFE-LO. Our proposed method’s accuracy may increase by simply performing hyperparameter selection during the elimination process rather than solely during pre-elimination. We also point out that the results herein for some previously published MFE-based methods, including MFE-LO, are the result of a more proper experimental evaluation of those particular methods. Lastly, we note it is our intention to compare in the future our method’s performance with additional methods, such as sparse estimation methods, e.g. for the case of linear classification the state-of-the-art Lasso and l1 logistic regression. For the case of nonlinear kernels, however, our proposed method may potentially have an advantage e.g. for some data domains, especially if the hyperparameter tuning benefits are sought; for the nonlinear case, we present extensive results obtained with our proposed method.

Keywords: SVM, feature selection, feature elimination, MFE, RFE, biomedical, biomarker

1 Introduction, Background, Related Work

Subset selection draws much interest; given an initially M -dimensional – often high-dimensional – dataset, the aim is to select a subset of this initial feature set, for purposes such as: 1) identifying a small subset of features – i.e. “markers” – necessary for making good predictions, especially important when M is huge and the number of samples is small; e.g. “biomarkers” for biomedical imaging studies e.g. [1, 2, 7, 15, 24] and gene studies in bioinformatics e.g. [11, 18]; 2) combatting the curse of dimensionality (COD) i.e. the potential degradation of generalization accuracy when the initial feature dimensionality M is increased beyond a certain point [21]; 3) reducing the complexity of the classification operation – both memory storage and computation – since accuracy gains attainable despite COD by a *large* feature set may be *small*.

Since there are $2^M - 1$ possible subsets, exhaustive search is practically prohibitive for even modest M e.g. less than thousands. Subset selection methods, which are thus heuristic, include “front-end” (aka “filtering”) methods, “wrapper” methods, and “embedded” methods [10, 13]. To briefly discuss these three categories: 1) *Front-end methods*: Some of these methods evaluate discrimination power of *small* feature groups prior to classifier training to combine small groups to form a final (retained) feature subset. Although this is robust to overfitting, it tends to not achieve *sufficiently high* generalization accuracy because features that can significantly improve generalization accuracy when taken with other features become ignored [10, 13].¹ Past work on front-end methods includes e.g. [16, 17]. 2) *Wrapper methods* include forward selection, backward elimination, and bidirectional methods; these methods repeatedly evaluate the classification accuracy of candidate feature subsets via *i*) classifier training (e.g. recursive feature elimination (RFE) method used in e.g. [11, 18], MFE methods such as MFE-LO and MFE-Slack [1]) or *ii*) without training the classifier in the reduced space (e.g. the ‘basic MFE’ method [1]); herein we will compare our results with both of these subcategories. Other past work on wrapper methods includes e.g. [25]. 3) *Embedded methods* aim to suppress irrelevant features via formulating the classifier training’s optimization problem in a way that *e.g. drives to zero* the values of many features whereby the rest of the features form the retained set. To that end, different norms (e.g. ℓ_1 , ℓ_2) and optimization approaches have been investigated, e.g. [20], [26], [9], [14], [4]. A premise of such embedded methods is the “bet on sparsity” principle for high-d data [20] which is essentially as follows: if the number of features far exceeds the number of samples and the true (and unknown) “weights” for the features are Gaussian, neither ℓ_1 nor ℓ_2 will estimate the weights well, due to the data being too little for estimating these nonzero weights due to COD, but solution sparsity can nevertheless be encouraged via use of ℓ_1 (whereby numerous features will be driven to zero).²

1.1 Brief review of SVMs

Consider a labeled training set $\{(\mathbf{x}_n, y_n)\}$, $n \in \mathcal{N} \equiv \{1, \dots, N\}$, where the n -th sample $\mathbf{x}_n = [x_{n,1} \dots x_{n,M}]^T \in \mathbb{R}^M$ has class label $y_n \in \{\pm 1\}$. For subscripted vectors, e.g. \mathbf{x}_n , the i -th coordinate value $x_{n,i}$ is also denoted \mathbf{x}_{ni} . A hyperplane acting as a binary (two-class) decision function is defined by $f(\mathbf{x}) \equiv \mathbf{w}^T \mathbf{x} + w_0$, $\mathbf{w} \in \mathbb{R}^M$, $w_0 \in \mathbb{R}$. Denoting $g_n \equiv g(\mathbf{x}_n) \equiv y_n f(\mathbf{x}_n)$ and $\mathbf{g} \equiv [g_1 \dots g_N]^T$, the signed distance from a data point \mathbf{x}_n to the decision boundary is $\frac{g_n}{\|\mathbf{w}\|}$. The decision boundary is a *separating* one if $g_n > 0 \forall n$, and its margin is accordingly defined as $\gamma \equiv \frac{\min_n g_n}{\|\mathbf{w}\|}$. An SVM is a linear or generalized linear two-class classifier that learns a separator for the training set with *maximum* margin. The “support vectors”, denoted by set $S = \{\mathbf{s}_1 \dots \mathbf{s}_T\}$ (with index set $\mathcal{S} = \{1, 2, \dots, T\}$), used to specify the SVM solution, are a subset of the training points at margin distance to the decision boundary. In the linear case, the SVM weight vector is given by $\mathbf{w} \equiv \sum_{k \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} \mathbf{s}_k$, where $\lambda_{\mathbf{s}_k}$ are scalar Lagrange multipliers. In the generalized linear (nonlinear) case, denoting $\underline{\phi}(\mathbf{x}) \equiv [\phi_1(\mathbf{x}), \dots, \phi_L(\mathbf{x})]^T$ where $\phi_i(\mathbf{x})$ are nonlinear functions of the \mathbf{x} coordinates, inner products between $\underline{\phi}(\mathbf{x})$ and $\underline{\phi}(\mathbf{u})$ that can be efficiently computed via a positive definite kernel function $K(\mathbf{x}, \mathbf{u}) \equiv \underline{\phi}^T(\mathbf{x}) \underline{\phi}(\mathbf{u})$ are of particular interest; in this case both $\underline{\phi}(\cdot)$ and \mathbf{w} itself need not be explicitly defined since both the SVM discriminant function $f(\cdot)$ and the SVM weight vector squared 2-norm can be expressed solely in terms of the kernel, *i.e.*:

$$f(\mathbf{x}) = \sum_{k \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} K(\mathbf{s}_k, \mathbf{x}) + w_0, \quad (1)$$

$$\|\mathbf{w}\|^2 = \sum_{k \in \mathcal{S}} \sum_{l \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} \lambda_{\mathbf{s}_l} y_{\mathbf{s}_l} K(\mathbf{s}_k, \mathbf{s}_l). \quad (2)$$

¹Note that a feature useless by itself for class separation (e.g. producing fully overlapping class-conditional densities) can improve accuracy when taken with other features, and two useless features can be useful together [10, 13].

²Embedded methods estimate how a cost function (e.g. an objective function in an optimization problem formulation) will change with movements in a feature subspace – often performed in a backward or forward framework, this approach is able to produce nested subsets of features [10].

This approach (the “kernel trick”), where $K(\cdot, \cdot)$ is explicitly specified and provided to the SVM training, is known as the “nonlinear kernel case”.

For linear or nonlinear kernel case, the basic SVM training problem is:

$$\min_{\mathbf{w}, w_0} \frac{1}{2} \|\mathbf{w}\|^2 \quad s.t. \quad y_n f(\mathbf{x}_n) \geq 1, \quad \forall n \in \mathcal{N} \quad (3)$$

Recall that $f(\mathbf{x}_n)$ in (3) simply stands for $\mathbf{w}^T \mathbf{x}_n + w_0$ in the linear case and $\langle \mathbf{w}, \underline{\phi}(\mathbf{x}_n) \rangle + w_0$ in the nonlinear kernel case; *i.e.*, note that \mathbf{w} and w_0 are a part of the constraints in (3) (even though above they do not appear so, due to (3) having been conveniently stated generically to cover both linear and nonlinear cases). The relationship of (3) to margin maximization can be understood as follows [1]. Assuming we have a separator (*i.e.* the training data is separable), with margin γ equal to $\frac{\min_n g(\mathbf{x}_n)}{\|\mathbf{w}\|}$, note that $g(\cdot) \equiv yf(\cdot)$ can be amplitude-scaled by an arbitrary nonzero constant ρ without altering the decision boundary. In particular, by forming $\tilde{g} = \rho g$, where $\rho = \frac{1}{\min_n g(\mathbf{x}_n)}$ so as to make $\min_n g(\mathbf{x}_n)$ equal to 1 (for consistency with the constraints), the margin γ is given by $\frac{\min_n g(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$. We can thus see the well-known result that, for this special choice of ρ , maximizing margin is equivalent to minimizing the squared weight vector 2-norm. The SVM training problem can alternatively be posed as

$$\min_{\mathbf{w}, w_0, \underline{\xi}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n \quad s.t. \quad \xi_n \geq 0, \quad y_n f(\mathbf{x}_n) \geq 1 - \xi_n, \quad \forall n \in \mathcal{N} \quad (4)$$

so as to allow slackness ($\underline{\xi} \equiv [\xi_1, \dots, \xi_N]^T$) in the margin constraints. (4) allows some support vectors to be practically closer than others to the hyperplane (by nonnegative slackness amounts ξ_n), thus handling both margin violations (*i.e.*, $\xi_n > 0$) and nonseparable data (a classification error occurs for sample n if $\xi_n > 1$).³ For choosing the SVM training parameter C (and other SVM hyperparameters in the nonlinear kernel case), the standard practice of using a validation or cross-validation procedure [8, 12] can be employed. The relationship of (4) to margin maximization can be understood as follows [1]. Notice in (4) that if C is made sufficiently large, no margin slackness will be tolerated and minimizing (4) *reduces to* minimizing the squared weight vector 2-norm and, thus, to maximizing margin. We thus see that (4) is a generalization of strict margin maximization (3) that *specializes* to strict margin maximization when C is made sufficiently large.

1.2 Related work on SVM-based feature elimination

Relating the SVM concepts in Sec. 1.1 to feature elimination algorithms, [1] proposed both a method called “basic MFE” that eliminates the feature whose elimination preserves maximum (positive) margin in the reduced space, and a second, analogous method called “MFE-slack” that eliminates the feature whose elimination yields the smallest SVM objective function (4) in the reduced space. That is, in accordance with the discussion above in Sec. 1.1, the theoretical basis for ‘basic MFE’ is *strict margin maximization*, and the theoretical basis for MFE-slack is the *generalization of strict margin maximization*, with the two methods being analogous, as described above and in [1].

To now re-state these two feature elimination methods presented in [1], let \mathcal{R} denote the retained features set, which is initially the full set of available features to eliminate from and becomes smaller during the process of eliminating features, and let $(\mathbf{w}^{\mathcal{R}}, w_0^{\mathcal{R}})$ denote the associated SVM weights. Let q^{-m} (or alternately $q^{\mathcal{R} \setminus m}$) denote quantity q upon (candidate or actual) elimination of feature m , with q^{-m, n_a} additionally indicating a choice of “anchor sample” n_a discussed below. As given by [1], the feature m^* eliminated by ‘basic MFE’ is

$$m^* = \arg \max_{m \in \{\tilde{m} \in \mathcal{R} | g_l^{-\tilde{m}} > 0 \quad \forall l \in \mathcal{N}\}} \min_n g_n^{-m} / \|\mathbf{w}^{-m}\| \quad (5)$$

³Note that \mathbf{w} and w_0 are a part of constraints (3) and (4) even though they do not appear there.

and, *similarly*, the feature m^* eliminated by MFE-Slack (when paired with “anchor sample” n^* discussed next) is

$$(m^*, n^*) = \arg \min_{m \in \mathcal{R}} \min_{n_a \in \{l \in \mathcal{N} | g_l^{-m} > 0\}} \frac{1}{2} \|\mathbf{w}^{-m}\|^2 (\rho^{-m, n_a})^2 + C \sum_{n=1}^N \xi_n^{-m, n_a} \rho^{-m, n_a} \quad (6)$$

As described in [1], this is a discrete optimization wherein, for each *candidate* feature for elimination, m , *every* (correctly classified) sample is evaluated as the potential margin-setting sample n_a (dubbed “anchor sample”), with both the weight vector 2-norm and the slackness values evaluated *post*-feature-elimination⁴, and subsequently the optimal (feature, sample) pair (m^*, n^*) that, post-elimination, minimizes (4) over all the discrete choices $\{(m, n_a)\}$ is selected, identifying the feature m^* to be eliminated.

Note that ‘basic MFE’ can only be used when data is separable, whereas its *counterpart* MFE-slack is not only consistent with margin maximization central to SVM learning but also deals with the case of nonseparable data by incorporating margin slackness into the elimination criterion as done by the standard SVM formulation (4). There is also the “hybrid” method – introduced in [1] and referred to herein as MFEhybrid – which is defined such that ‘basic MFE’ is used at the elimination steps the data is separable and MFE-Slack is used at the other steps (when data is nonseparable).

The third feature elimination approach proposed in [1] is the LO (Little Optimization) approach; an MFE variant which, like ‘basic MFE’, requires the data to be separable upon candidate elimination of a feature. The premise of LO is that one can employ a *type* of classifier retraining with little (exceptionally modest) computation to obtain a margin guaranteed to be larger than the margin that ‘basic MFE’ can obtain alone. Specifically, given SVM weights (\mathbf{w}, w_0) in the reduced space (upon candidate elimination of a feature), LO considers the *new* parameterized weight vector $(A\mathbf{w}, w_0)$, where A and w_0 are *scalar* parameters to be optimized, with \mathbf{w} held fixed. That is, allowing adjusting the squared weight vector 2-norm and the intercept w_0 (with the weight vector orientation fixed), LO poses the standard SVM training problem but optimizing only in this *two*-dimensional (A, w_0) parameter space, stated as (7) for the linear kernel case and as (8) for the nonlinear kernel case [1]:

$$\min_{A, w_0} A^2 \text{ s.t. } y_n(A(\mathbf{w}^T \mathbf{x}_n) + w_0) \geq 1, \forall n \in \mathcal{N} \quad (7)$$

$$\min_{A, w_0} A^2 \text{ s.t. } y_n(A(\sum_{k \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} K(\mathbf{s}_k, \mathbf{x}_n)) + w_0) \geq 1, \forall n \in \mathcal{N} \quad (8)$$

It was discussed in [1] that one can perform candidate elimination of each feature and then perform LO in the reduced space, so as to then pick the candidate elimination that leads to the largest post-LO margin. This elimination method, where LO is embedded into the elimination decision, is referred to as MFE-LOemb herein.

Next we propose a new novel feature elimination method in Sec. 2 where we also discuss limitations of the above earlier MFE methods as motivation for our proposed method. Based on the background provided by Sec. 2, we then propose a second method in Sec. 3 which performs better.

2 MFE-QP: new feature elimination method

Given SVM linear weights (\mathbf{w}, w_0) , we consider the *new* parameterized weight vector $(a\mathbf{w}, b)$, to jointly optimize the scalar parameters a , b , and N slacknesses, with \mathbf{w} held fixed. We thus pose

⁴To arrive at (6) from (4) under such consideration (*i.e.*, upon candidate elimination of feature m and in accordance with a particular choice of anchor sample n_a), one calculates the value ρ^{-m, n_a} as $1/g_{n_a}^{-m}$ (so that $\rho^{-m, n_a} g_{n_a}^{-m}$ will be 1 as discussed by the ρ discussion in Sec. 1.1), based on which the square of the weight vector 2-norm and slackness values are then measured and plugged into the objective function (4) (*i.e.* the squared 2-norm measurement $\|\mathbf{w}^{-m}\|^2 (\rho^{-m, n_a})^2$ shown in (6) and the slackness measurement $\max(0, 1 - \rho^{-m, n_a} g_{n_a}^{-m}) \forall n$ which is the “ $\xi^{-m, n_a} \rho^{-m, n_a}$ ” shown in (6)).

the standard slackness-incorporating SVM problem (4) consistent with margin maximization but only optimize in this $(a, b, \underline{\xi})$ parameter space:

$$\min_{a, b, \underline{\xi}} \frac{1}{2} a^2 \|\mathbf{w}\|^2 + C \sum_n \xi_n \text{ s.t. } \xi_n \geq 0, y_n(a \mathbf{w}^T \mathbf{x}_n + b) \geq 1 - \xi_n, \forall n \in \mathcal{N} \quad (9)$$

For the nonlinear kernel case, where \mathbf{w} need not be explicitly defined as discussed in Sec. 1.1, the problem (9) would be written as (10) equivalently; *i.e.*, the sum $\mathbf{w}^T \mathbf{x}_n$ is written as $\sum_{k \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} K(\mathbf{s}_k, \mathbf{x}_n)$.

$$\min_{a, b, \underline{\xi}} \frac{1}{2} a^2 \|\mathbf{w}\|^2 + C \sum_n \xi_n \text{ s.t. } \xi_n \geq 0, y_n(a(\sum_{k \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} K(\mathbf{s}_k, \mathbf{x}_n)) + b) \geq 1 - \xi_n, \forall n \in \mathcal{N} \quad (10)$$

At first look, our formulations (9) and (10) may seem computationally costly and complicated, but they are in fact each equivalent to a very simple problem with little computational cost – the 1-dimensional SVM – as follows. Since \mathbf{w} is an input held fixed (*i.e.* it is not one of the *parameters* optimized by the problem) and $\|\mathbf{w}\|$ is only a fixed scalar (and not one of the parameters optimized by the problem), we can define a new (scalar) *parameter* $w \equiv a \|\mathbf{w}\|$ as a replacement for parameter a and new (scalar) data variables $z_n \equiv \frac{\mathbf{w}^T \mathbf{x}_n}{\|\mathbf{w}\|}$ as replacement for data variables \mathbf{x}_n .⁵ With this change of variables, each of (9) and (10) can be rewritten as follows.

$$\min_{w, b, \underline{\xi}} \frac{1}{2} w^2 + C \sum_n \xi_n \text{ s.t. } \xi_n \geq 0, y_n(w z_n + b) \geq 1 - \xi_n, \forall n \in \mathcal{N} \quad (11)$$

Notice that (11) is simply a 1d SVM (4); the scalars w and b together define the 1d SVM decision boundary *i.e.* a 1d threshold. Our proposed approach of (9) and (10) – subsequently equivalently written as (11) – is named “QP”. Since the QP approach requires very little computation it can be performed in conjunction with each feature elimination step. Specifically, given a current retained feature set \mathcal{R} , we can perform candidate elimination of each feature $m \in \mathcal{R}$, train the 1d SVM (11) in the reduced space (*i.e.* under candidate retained feature set $\mathcal{R} \setminus \{m\}$), and then pick the candidate elimination that leads to the smallest SVM objective function value (11) in the reduced space. This embedding of QP into the elimination decision – consistent with margin maximization central to SVM learning – defines our proposed feature elimination method, named MFE-QPemb, given by (12) below; *i.e.* the feature m^* proposed to be eliminated is:

$$m^* = \min_{m \in \mathcal{R}} \min_{w, b, \underline{\xi}} \frac{1}{2} w^2 + C \sum_n \xi_n \text{ s.t. } \xi_n \geq 0, y_n(w z_n^{-m} + b) \geq 1 - \xi_n, \forall n \in \mathcal{N} \quad (12)$$

Notice that the only computation required by this method is to – for each candidate feature m for elimination – compute the 1d (scalar) data points z_n^{-m} ($\forall n \in \mathcal{N}$) from training points \mathbf{x}_n (using recursion, to reduce computation), then train a 1d SVM with these scalar training points z_n^{-m} to obtain the $(w, b, \underline{\xi})$ triplet; and lastly plug these obtained $(w, b, \underline{\xi})$ values into (12) to determine the winning candidate m^* .

Rather than solving the 1d SVM (11) in the usual way (by using a “dual” solver), an SVM training algorithm specifically designed for the 1d case for significantly reduced computation can be utilized; this algorithm with $O(n \log n)$ complexity (the complexity of sorting n numbers), given in Appendix 6, is only for computation reduction for the 1d SVM training step and is thus recommended but not *required per se* by our proposed QP approach.

As discussed in Sec. 1.2, a previous method with properties similar to our proposed method is MFE-Slack [1]. MFE-Slack is similar due to performing SVM feature elimination that is both consistent with margin maximization and incorporates margin slackness. However, a limitation of MFE-slack is that it makes no alteration to the decision boundary that results solely from the

⁵In the nonlinear kernel case, we define z_n instead as $z_n \equiv \sum_{k \in \mathcal{S}} \lambda_{\mathbf{s}_k} y_{\mathbf{s}_k} K(\mathbf{s}_k, \mathbf{x}_n) / \|\mathbf{w}\|$.

removal of the eliminated feature⁶; it is based on *discrete optimization*, whereas our proposed approach QP performs *non-discrete optimization* – similarly with only little computational cost, in the form of a *1d SVM* – altering the decision boundary so as to obtain a lower, more optimal SVM objective function value than MFE-Slack. That is, essentially our proposed approach seeks to produce *modestly more margin maximization than MFE-Slack at each feature elimination step*, within the standard slackness-based SVM nonseparability framework (4) that both methods are based on. For illustrative purposes, in Fig. 1, on the 2000-feature *Colon Cancer* gene dataset representative of high-d datasets, average test set classification error rate (an average across “trials”⁷) is plotted, as a function of the number of retained features (which is reduced going from right to left), for the nonlinear Gaussian kernel as well as the polynomial kernel.⁸ The Figure illustrates that our proposed method MFE-QPemb achieved much higher generalization accuracy than MFE-Slack and MFEhybrid. The other curves in the plot will be discussed shortly.

Another motivation for our proposed QP approach is that it enables to *avoid* “full (SVM) retraining” (“FR”, *the recalculation of all SVM weights from scratch*) at individual feature elimination steps; *i.e.*, instead of full SVM retraining, one can perform QP which similarly is a *type* of classifier retraining.⁹ Avoiding FR for high-d datasets is important for several reasons. First, if FR is performed stepwise (for the sake of stepwise producing *optimal SVM quantities* *i.e.* higher margin, or smaller SVM objective function value, than obtained by the above methods that do not employ FR), the overall feature selection process may, as we will evaluate herein, suffer from “overfitting”, *i.e.* the “overlearning” of data, which is a limitation. This is especially an issue for *high-dimensional data* since overfitting – *a cumulative effect* – is expected when the number of feature elimination steps accumulating is *very large*. To illustrate this, also included in Fig. 1 are results for MFE-Slack-FR_{sub} (a method where FR is performed at each feature elimination step *subsequent* to the feature elimination decision by MFE-Slack) and similarly MFEhybrid-FR_{sub} (*i.e.* MFEhybrid’s FR-based counterpart similarly). Notice from the results of these two FR-based methods that, while the use of FR has brought an increase in generalization accuracy, these methods that are employing *stepwise full SVM retraining throughout a very large number of elimination steps* are achieving less generalization accuracy than our proposed method MFE-QPemb (which has the additional benefit of having less computational cost, as we will discuss next); this result can perhaps be understood as a type of overfitting. To illustrate that there may not be much overfitting for – by contrast – a *relatively low-dimensional dataset*, we give Fig. 2, where we demonstrate that the above FR-based methods are now *outperforming* our proposed method MFE-QPemb, unlike in the high-d case of Fig. 4 above.¹⁰ Second, FR is computationally much more costly than our proposed QP approach. If the initial feature dimensionality is M (e.g. 7000 or more, for gene data), at the i -th elimination step FR will need to *train an SVM for a very large data dimensionality $M - i$* (such as 6999, 6998, ...), whereas our proposed method *only trains a 1d SVM* *i.e.* the number of dimensions is *1 throughout elimination steps*.

Moreover, due to incorporating slackness, QP does not require the data to be separable at a feature elimination step. By contrast, the previously proposed similar MFE-LOemb method (based on the similar LO approach), described in Sec. 1.2 above, requires separability. Requiring

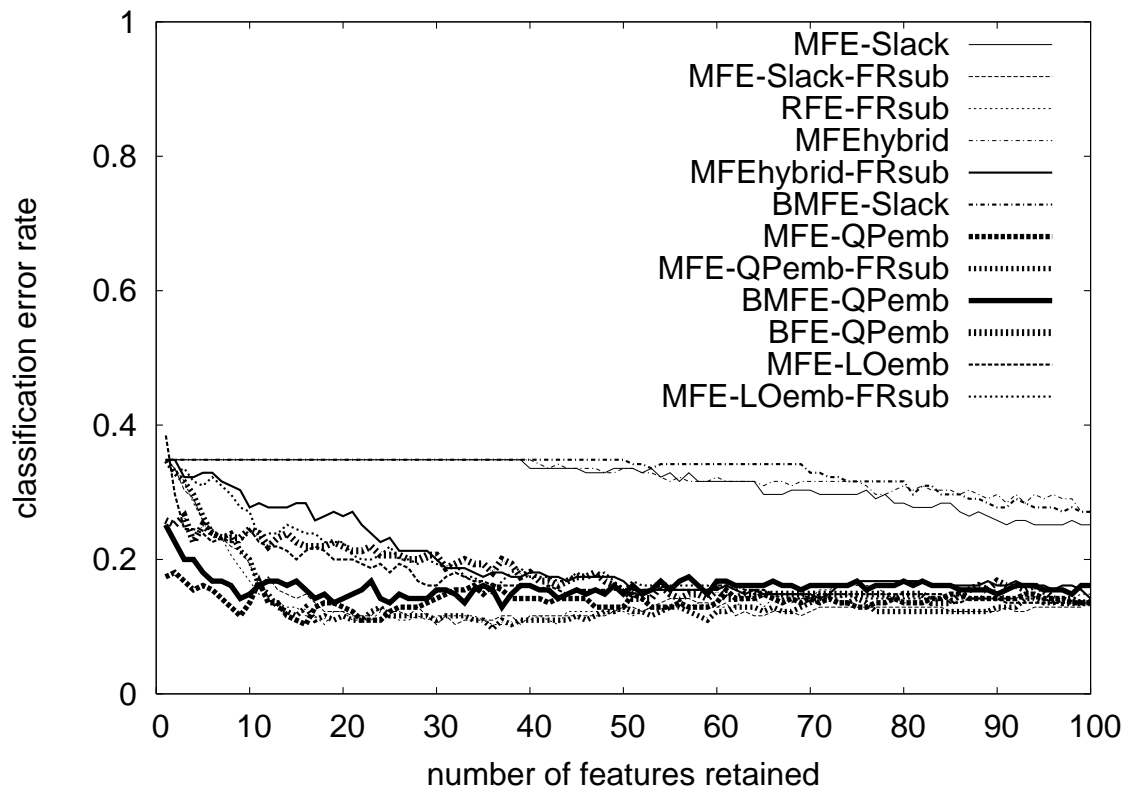
⁶That is, throughout elimination steps, relative magnitudes within the set of all originally designed *i)* SVM weights including the hyperplane intercept w_0 , or equivalently, *ii)* Lagrange multipliers, remain unchanged.

⁷A “trial”, defined in Sec. 4 and summarized here, is a random 50 – 50% split of the dataset into a held-out set and a non-heldout set. To be used as initial classifier by all elimination methods, an SVM is trained on the entire non-heldout set, using SVM hyperparameter values determined by 5-fold cross-validation on this set. Feature elimination is then performed, with error rate measured on the held-out (test) set.

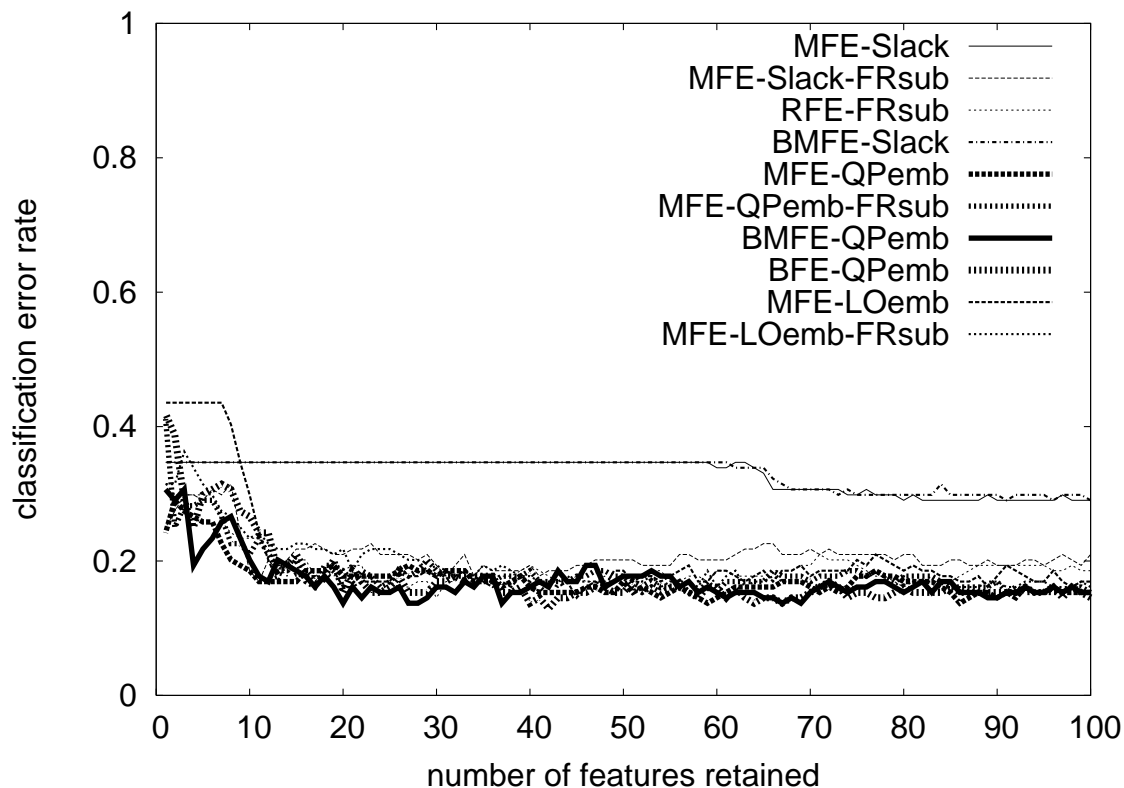
⁸Herein, in graphs we zoom in on the end (*i.e.* the most interesting) segment of the elimination process, since methods’ performance relative to each other does not significantly vary during the beginning segment not shown.

⁹Note also that the use of QP does not preclude subsequent use of FR. After the feature elimination decision is made using QP, one can opt to use, or not use, full SVM retraining subsequently, before proceeding to eliminate another feature.

¹⁰In this particular comparison, MFE-QPemb serves a reference role due to being present in both (low-d and high-d) Figures.



(a) Gaussian kernel.



(b) Polynomial kernel.

Figure 1: Average test set classification error rate for the high-d *Colon Cancer* gene dataset, for (a) the Gaussian kernel and (b) the polynomial kernel.

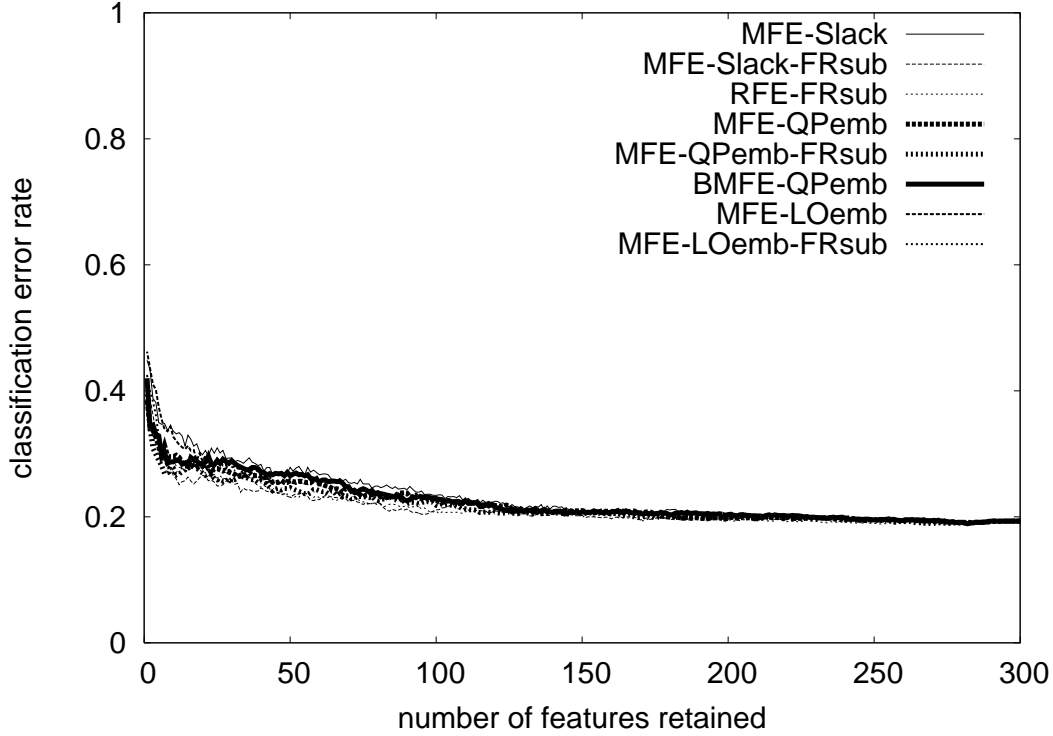


Figure 2: Average test set classification error rate for the somewhat high-dimensional *UCI Arcene* dataset (where we only used 300 initial features of this dataset), for the Gaussian kernel.

separability, i.e. strictly satisfying the margin, may lead to overfitting when training samples at the margin are outliers or even *mislabeled* samples [1]. For illustrative purposes, Fig. 1 also includes results for the MFE-LOemb method and its FR-based counterpart MFE-LOemb-FRsub (which performs FR at each feature elimination step *subsequent* to the feature elimination decision by MFE-LOemb). Notice that both of these LO-based methods are outperformed by our proposed method MFE-QPemb that does not utilize FR, for both nonlinear kernels used. Again, the results presented here in this Section serve illustrative purposes; more extensive evaluations are given in the Results section (Sec. 4).

Notice that hyperparameter C is a part of our proposed method’s elimination criterion (12), and thus hyperparameter tuning can be incorporated into the method’s elimination decision. Herein we do not; i.e. the only hyperparameter tuning we perform is cross-validation *during pre-elimination* to obtain (train) the initial SVM in the original full feature space (4). In future work, *tuning additionally during elimination* may bring an increase to the generalization accuracy of our proposed QP approach, without bringing an unacceptable amount of increase in complexity (both computation and memory storage; especially if this tuning is solely batch-performed i.e. upon each elimination of a batch of features rather than each feature). By contrast, the LO approach – by nature of its definition which does not include C or a similar hyperparameter – does not allow incorporating hyperparameter tuning into the elimination decision; this is another limitation of the LO approach. Herein we will show that our proposed approach QP achieves higher or competitive generalization accuracy performance (lower or similar test set classification error rate) compared with LO *despite not utilizing the option to tune hyperparameters during elimination*.

Above we described and illustrated – for SVM-based “backward feature elimination” – that adjusting the current solution in the reduced feature space (upon candidate elimination of a feature), by either a light retraining of the classifier (via e.g. QP or LO) or full SVM retraining (FR), brought an increase to generalization accuracy; i.e. performed better than ‘basic MFE’ (or MFEhybrid) and MFE-slack. Clearly, there is a discrepancy between this finding and [1]’s earlier finding that stepwise classifier retraining did not improve accuracy of the MFE-based feature elimination methods. The explanation is simple: not for the first, *non-retraining* group of methods (‘basic MFE’, MFE-slack,

and their hybrid MFEhybrid) in [1] but rather only for the second, *retraining* group of methods in [1] (“MFE-Retrain” (aka MFEhybrid-FR), and MFE-LO), it turns out that test set accuracy evaluation was performed less properly in [1] than herein. Specifically, in [1], when the elimination sequence – after its *entirety* was obtained – was applied to the test set (i.e. a *retroactive* application of a *complete* elimination sequence to a test set), the test set’s features were removed in accordance with this correct sequence, while each stepwise boundary, used to measure test set error rate, was the boundary that results *solely* from removing that step’s eliminated feature. By contrast, herein the test set error rate was measured more properly, on post-retraining boundaries propagated across the steps of the elimination process that is being applied to the training set. In other words, essentially the results we present here for “full (SVM) retraining (FR)” and LO [1] are more proper results than the results presented for these two approaches earlier in [1].

3 BMFE: feature elimination that combines “expected risk” bound and margin maximization

The second novel feature elimination method we propose herein, named BMFE-QP, *also intended primarily for use with high-d data*, is based on combining the premise of a well-known theoretically motivated upper bound (by Vapnik et al.) on “expected risk” (of making a classification error) in Statistical Learning Theory [23, 12] with the premise of the SVM-based QP feature elimination approach we proposed in Sec. 2. We begin with an introductory discussion on the bound in Sec. 3.1 and discuss our proposed method in Sec. 3.2.

3.1 Background on “expected risk” bound

Statistical Learning Theory [22, 23, 12] is concerned with producing a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that estimates from an input $\mathbf{x} \in \mathcal{X}$ an output $\mathbf{y} \in \mathcal{Y}$ considered to be the “truth” associated with \mathbf{x} . For instance, such estimation may be 2-class classification, with the sought function being a hyperplane (i.e. the decision boundary) as herein, and \mathbf{x} a vector $\mathbf{x} \in \mathbf{R}^M$ with class label $y \in \{\pm 1\}$. More precisely, a “learning machine” is defined by a family of possible mappings $\mathbf{x} \mapsto f(\mathbf{x}, \Theta)$ parameterized by Θ , with “learning” (aka training) being the step of employing a “training set” X of N input samples in order to choose a particular Θ value whereby one is simply left with the desired mapping $\mathbf{x} \mapsto f(\mathbf{x})$. Θ are the “hyperparameters” – the tuning parameters – whose values are commonly set (tuned) by a validation procedure such as cross-validation (used in our work herein) [8, 12]; *e.g.*, in the SVM linear kernel case, the only hyperparameter is C (4).

The VC dimension h is a property of a family of functions parameterized by Θ ; it is a measure of the learning machine’s “capacity”, which refers to its ability to learn any training set without error [3, 12]¹¹, which, in our work herein on classification, means no classification errors. For a learning machine with VC dimension h , the bound (13) on “expected risk” $R(\Theta)$ holds with (high) probability $1 - \eta$ (where $0 \leq \eta \leq 1$ can be chosen arbitrarily small) [12, 22].

$$R(\Theta) \leq R_{emp}(\Theta) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}}. \quad (13)$$

$R_{emp}(\Theta)$ in (13), given by $\frac{1}{2N} \sum_n |y_n - f(\mathbf{x}_n, \Theta)|$, is the “empirical risk”, the mean error rate measured on the training set.¹² Since $R_{emp}(\Theta)$ is a fixed number for a given choice of training set $\{(\mathbf{x}_n, y_n)\}$ and the Θ chosen by the training, the right-hand side of (13) can be calculated if one knows h [3]. A principled way of choosing a learning machine is to choose from among a set of candidates the one with the lowest value for the right-hand side of (13) [3, 12]. To minimize the right-hand side, there is a tradeoff between minimizing the first term (the training error) and

¹¹VC dimension examples, including examples discussing it in the context of a function family’s number of parameters, can be found in [3].

¹²This particular way of writing $R_{emp}(\Theta)$ assumes the loss function $L(u, v)$ is 0 if $u = v$ and is 1 otherwise; *e.g.*, if $\mathcal{Y} = \{\pm 1\}$ (used commonly and in our work herein), $L(u, v)$ is $\frac{1}{2}|v - u|$.

the second term (called the “VC confidence (VCC)” term) depending on the complexity of the machine through the machine capacity measure h . The candidates can be evaluated according to the Structured Risk Minimization (SRM) induction principle, defined to introduce a “structure” into the candidate set F^{cands} by considering nested subsets $F_1 \subset F_2 \subset \dots \subset F_Z$ (for some integer Z , with $F_Z \subseteq F^{cands}$) with associated known VC dimensions $h_1 < h_2 < \dots < h_Z$ (or known bounds on the VC dimensions). In this way, for a given subset F_i , the goal would be to minimize R_{emp} among members of that subset. Accordingly, upon training a number of machines across subsets (possibly as few as simply one machine per subset), one can simply choose the machine with the least sum of R_{emp} and VCC. SVM was “one of the first practical learning procedures for which useful bounds on the VC dimension could be obtained and hence the SRM program could be carried out” [12]. Specifically, SVM produces a hyperplane decision function $f_{\mathbf{w},b} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ ¹³ defined on training points (or their basis expansions, for the nonlinear kernel case) contained in a ball of radius r ¹⁴, and, for some scalar A , the VC dimension h of the function family $\{f_{\mathbf{w},b} : \|\mathbf{w}\| \leq A\}$ has the following bound¹⁵[12, 22]:

$$h \leq r^2 A^2 + 1 \quad (14)$$

That is, as well-known, the standard SVM formulation (4) is a *specific* implementation – an instantiation – of the SRM principle for minimizing the (*generic*) bound (13); C serving to achieve a *tradeoff* between $\sum_n \xi_n$ and $\|\mathbf{w}\|^2$ in (4) is an instantiation of seeking the tradeoff discussed above between the two terms of (13).¹⁶ It is, however, important to recognize that (13) and (4) implement “tradeoff” in their own mathematically distinct ways; i.e. (13) is the *generic* approach, from which differs the *specific* (and mathematically distinct) (4). To see this, first notice that unlike C in (4) there is *no explicit dedicated tradeoff (i.e. tuning) parameter* in (13). Second, the VCC term of (13) and the $\|w\|^2$ term of (4) are mathematically quite different; e.g. (4) does not explicitly include the data radius r whereas (13) essentially does via (14) (because (14) can be used to rewrite (13) as (15) by simply plugging in the upper bound $r^2\|w\|^2$ for h).

$$R(\Theta) \leq R_{emp}(\Theta) + \sqrt{\frac{r^2\|w\|^2(\log(2N/r^2\|w\|^2) + 1) - \log(\eta/4)}{N}}. \quad (15)$$

Overall the bound (13) is simply a guide – among two machines achieving zero empirical risk (i.e. no classification errors in training set), it is possible for the one with the higher VC dimension

¹³This function is canonical i.e. there exists a “margin-setter” sample \mathbf{x} for which $y(\mathbf{w} \cdot \mathbf{x} + b)$ is 1, as stated in the constraints of the standard SVM formulation (4).

¹⁴One way the ball radius r can be defined is to define it as the maximum Euclidean distance between any two (training) points; i.e. $\max_{i,j \in \mathcal{N}} \|\mathbf{x}_i - \mathbf{x}_j\|$.

¹⁵Note, however, that the ball around the data points means that this approach to bounding the VC dimension depends on *observed values* of the features; i.e. as noted by [12] (p.389) “in a strict sense, the VC complexity of the class is not fixed a priori, before seeing the features”. It is, however, possible to gain insight as to why maximizing the margin, the SVM objective (via the standard slackness-based formulation (4)), is considered important for good generalization performance. To that end, note the family of SVM-like classifiers named “gap tolerant classifiers” discussed in [3] which are based on *both* the idea of putting balls around data points *and* hyperplanes – the classifier is specified by the location of a ball (in \mathbf{R}^M) and two parallel hyperplanes with parallel normal vectors (in \mathbf{R}^M), and the decision function classifies points as one of two classes so long as these points lie inside the ball but not between the hyperplanes (i.e., so long as the points are not members of the so-called “margin set” of points that may lie between the hyperplanes). The VC dimension of such a family of classifiers can be controlled by controlling both the maximum allowed ball diameter and the minimum allowed perpendicular distance between the two hyperplanes [3] – subsequently discussing along this line with examples, [3] argues that it seems very reasonable to conclude that SVMs too gain a similar kind of capacity control from their training due to their training objectives being very similar to gap tolerant classifiers’. The discussion above suggests that although a *rigorous* explanation for why SVMs often achieve good generalization performance is not provided by SRM alone there is clearly a theoretical connection between SVMs’ objective of margin maximization and SRM.

¹⁶Notice that the sum $\sum_n \xi_n$ in (4) corresponds to (13)’s R_{emp} ; it is an upper bound on R_{emp} (the number of classification errors), because, by definition of slackness, 1) each term in the sum is nonnegative, and 2) the sum contains a term for each classification error (with each such term > 1).

to achieve better generalization performance.¹⁷ Next we introduce a novel feature elimination method motivated by and consistent with the above theoretical details.

3.2 BMFE-QP: second new feature elimination method

In Sec. 3.1, we discussed the risk bound (13) and the theoretical connection between the SRM principle (for minimizing this bound) and the standard SVM formulation (4). Now let us relate this to feature elimination algorithms. In particular, we would like to craft a feature elimination criterion based on the bound and consistent with the well-established theory.

One feature elimination approach (the immediately obvious one) would be to eliminate the particular feature m^* whose elimination minimizes the bound (15):

$$m^* = \min_{m \in \mathcal{R}} R_{emp}^{-m} + \sqrt{\frac{r_m^2 \|\mathbf{w}^{-m}\|^2 (\log(2N/r_m^2 \|\mathbf{w}^{-m}\|^2) + 1) - \log(\eta/4)}{N}} \quad (16)$$

Although this feature elimination criterion (16) is consistent with theory, it is not crafted very well, as follows. Suppose for a candidate feature m for elimination we obtain the quantities R_{emp}^{-m} and $\|\mathbf{w}^{-m}\|^2$ via SVM and plug them into (16) to evaluate the bound (16). That is, upon having already tuned a good tradeoff between these quantities through the tradeoff parameter C for *specifically the SVM machine context* (without explicit use of data radius r_m), we would now be somewhat abandoning that tradeoff in favor of making an alternate tradeoff attempt via direct bound minimization (16) for the *generic machine context* which, by contrast, 1) makes explicit use of data radius r_m and 2) no use of C . Clearly, *from a numerical optimization standpoint*, this is essentially an “apples and oranges” situation, because the tradeoff suitability of these two particular values as a pair comes from having been numerically *fused* into a single measurement through optimization (i.e. the optimized SVM objective function measurement $\frac{1}{2}\|\mathbf{w}^{-m}\|^2 + C \sum_n \xi_n$ ¹⁸) via the fusing parameter C , whereas their use in (16) – which “unfuses” them and treats them as separate entities *yet under their fusion-derived values* – actually means both 1) use of these two quantities in a now numerically mismatched context *and* 2) loss of information obtained for hyperplane-specific tradeoff. It may be wiser to instead *preserve* the tradeoff that had already been achieved by the SVM, since that tradeoff is for the particular decision boundary type we are seeking in the reduced space: a hyperplane. That is, it may be wiser to craft a feature elimination criterion that combines the bound concept with the *whole (intact) form of the SVM objective function*; i.e., we shall treat the SVM objective function as a single, intact quantity, rather than separate its key quantities R_{emp}^{-m} and $\|\mathbf{w}^{-m}\|^2$ as seen in (16). Thus we propose as feature elimination criterion the product of the data radius squared r_m^2 and the SVM objective function; here the *particular* SVM objective function we propose for the criterion is the 1d SVM proposed by our QP approach (11), because, 1) as illustrated in Sec. 2 and will be shown in the Results section, among all previous MFE-based elimination approaches that incorporate slackness to support nonseparability the one that achieves the best generalization is our proposed QP approach, and 2) QP has little computational complexity since it merely involves *1d* SVM training.

$$m^* = \min_{m \in \mathcal{R}} r_m^2 \min_{w, b, \underline{\xi}} \left(\frac{1}{2} w^2 + C \sum_n \xi_n \right) \text{ s.t. } \xi_n \geq 0, y_n(wz_n + b) \geq 1 - \xi_n, \forall n \in \mathcal{N} \quad (17)$$

Our proposed feature elimination method (17), named BMFE-QPemb, is consistent with theory. At an elimination step, when the data is separable (under candidate feature elimination), the first term R_{emp} of the bound (13) is 0, and thus our proposed method (17) *reduces to* minimizing the (theoretically motivated) bound (13) itself; *i.e.*, our method, by minimizing $r^2\|w\|^2$, essentially

¹⁷As an aside, note that the bound is guaranteed not tight when the VCC, which monotonically increases with h , exceeds a threshold (which, clearly, would be relative to the maximum value chosen for the loss function); *e.g.*, for the 0-1 loss above, a reasonable threshold is 1. For example, for this case, [3] plotted VCC versus h/N for $N = 10,000$ and $\eta = 0.05$ as well as illustrated that VCC exceeds 1 for $h/N > 0.37$.

¹⁸Recall that $\sum_n \xi_n$ here is essentially R_{emp}^{-m} as described in Sec. 3.1.

minimizes h due to (14), and therefore minimizes the second (and only nonzero) term in the bound (13). If, on the other hand, at the elimination step the data is nonseparable (under candidate feature elimination), we are simply multiplying r^2 with the standard nonseparability counterpart of $\|w\|^2$ rather than $\|w\|^2$ itself; i.e. as very commonly done consistent with theory (as discussed in Sec. 1.1), we are utilizing the standard SVM objective function $\frac{1}{2}\|w\|^2 + C \sum \xi_n$ as an alternative to the standard SVM objective function $\frac{1}{2}\|w\|^2$ (where the latter strictly maximizes the margin whereas the former allows for slackness in the margin constraints as discussed in Sec. 1.1 and [1]). It should be clear from the above that BMFE-QPemb is consistent with both margin maximization (central to SVM learning) and the “expected risk” bound (central to the SRM principle); it is consistent with well-established theory.

Notice in (17) that r_m^2 appears on the *outside* of the SVM objective function (i.e. left of the QP expression for 1d SVM) since it only depends on the data. That is, our proposed elimination criterion (17) *only requires 1d SVM retraining* (in the same precise way as our earlier QP criterion (11)) *and subsequently a multiplication (by r_m^2)*.¹⁹ That is, the criterion only requires little computation, and thus can be performed in conjunction with each feature elimination step. Fig. 4, for the *Duke Breast Cancer* gene dataset with 7129 features, illustrates that a *large* gain in accuracy is achievable *by employing our second proposed method BMFE-QPemb instead of our first proposed method MFE-QPemb*, for both nonlinear kernels used herein.²⁰

Above we have been supporting that our proposed method BMFE-QPemb (17) is crafted better than the alternative method (16) which eliminates by direct bound minimization. We name this alternative the “BFE” method (as opposed to “BMFE”) and now for illustrative purposes we couple BFE with our QP approach and arrive at “BME-QPemb”; i.e. the particular approach we employ in order to provide the discussed quantities R_{emp}^{-m} and $\|\mathbf{w}^{-m}\|^2$ to the BFE approach (16) is the QP approach. Consistent with our comments for (16), the results for BME-QPemb in Fig. 1 illustrate that this method achieves *lower* generalization accuracy than our proposed method BMFE-QPemb (17).

Notice in Fig. 1 that it lastly also illustrates results for “BMFE-Slack”, where, as the name suggests, we are defining this method by simply combining the BMFE approach (of multiplying by square of the data radius) with the MFE-Slack criterion (instead of combining BMFE with the QP criterion); i.e. BMFE-Slack is simply (6) with r_m^2 put into its objective function as a multiplicative term. The results illustrate that this, unlike BMFE-QPemb, is resulting in a generalization performance not significantly better than MFE-Slack’s itself, *which re-emphasizes that our proposed QP criterion is much more effective than the MFE-Slack criterion*.

Our BMFE approach (of multiplying the SVM-based objective function by square of the data radius) can be combined, in future work, with the LO criterion (instead of combining BMFE with the QP criterion); this approach would have the limitations of the LO approach, described above.

4 Results and Discussion

We performed experiments to compare our proposed methods (MFE-QPemb and BMFE-QPemb) with every previously proposed MFE-based method [1]: 1) MFE-Slack; 2) its separability-requiring counterpart ‘basic MFE’; 3) their hybrid MFE/MFE-Slack, herein named MFEhybrid *i.e. the three best performers in [1] among MFE-based methods*; 4) MFE-LO; and their “FRsub” variants that perform stepwise “full (SVM) retraining (*FR*)” *subsequent* to the decision to eliminate a particular feature (as discussed in Sec. 2) e.g. MFE-Slack-FRsub; as well as 5) RFE-FRsub [11]²¹.

Our proposed methods can be compared with additional feature selection methods such as sparse estimation methods that have been researched extensively over the past several years, such as the state-of-the-art Lasso and ℓ_1 -based logistic regression. Although we leave this comparison

¹⁹Note that r_m^2 itself can be computed with little computation cost using recursion.

²⁰By contrast, these two methods perform *similarly* in our earlier illustrative Figure which was for the *Colon Cancer* gene dataset (Fig. 1).

²¹Since RFE [11] performs full (SVM) retraining (i.e. “FR”) *subsequent* to the feature elimination decision, herein we refer to RFE as RFE-FRsub.

for future work, we note that our proposed methods could potentially have an advantage over sparse estimation methods when nonlinear kernels are considered, even under our current method definitions herein *where we do not yet employ the potential benefit of hyperparameter tuning during the feature elimination steps themselves*, as we discussed above. For the nonlinear case, note that herein we provide extensive results for our proposed methods.

The experiment procedure we used for training of the *initial* SVM classifier used by all feature elimination methods evaluated herein was the following common one. The dataset is randomly split 50-50% into a non-heldout (training) set X and a heldout (test) set \bar{X} . Each such split defines one “trial”. 5-fold cross-validation is then performed on the non-heldout set, to select classifier hyperparameters for each trial, from amongst a candidate set of hyperparameter values. The classifier is then retrained for these hyperparameter values using all of X .

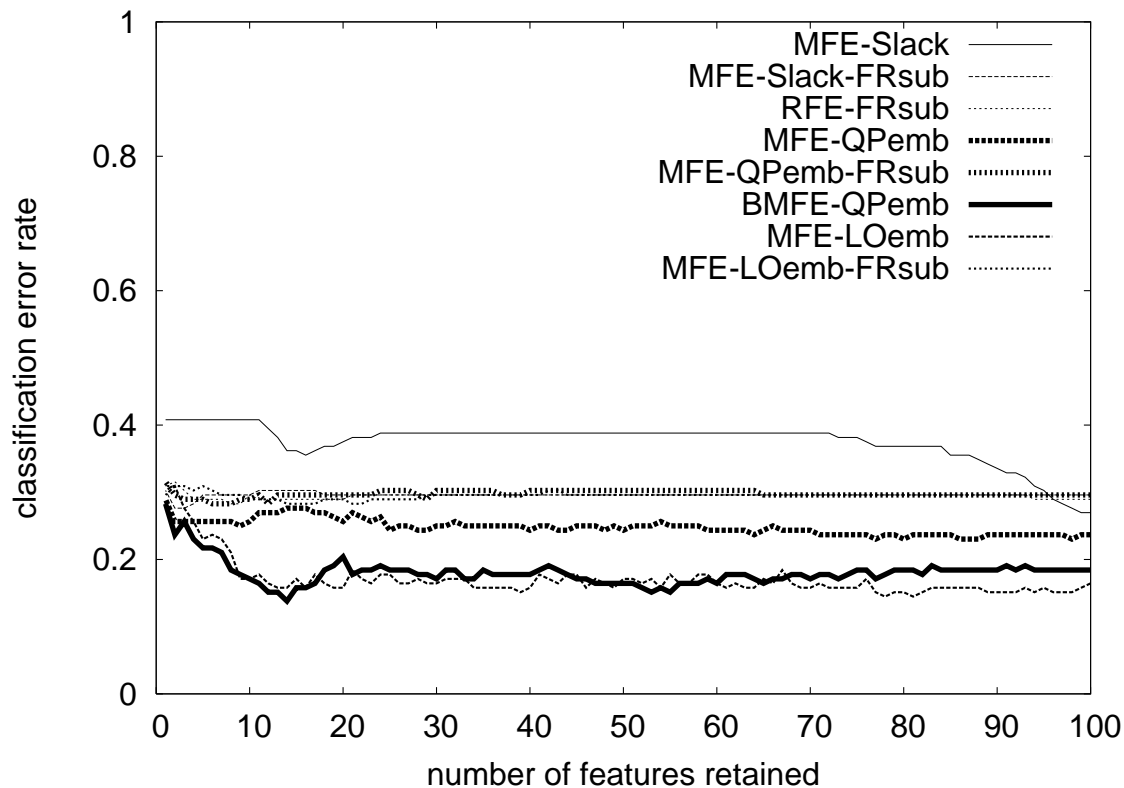
As discussed in [1], Cover’s linear dichotomy theorem [6] states that the probability that a training set (with points in general position) is linearly separable is very close to 1 when the number of samples is not bigger than the number of features plus one. For example, for the gene data and biomedical imaging data domains, typically there may be a huge number of features (*e.g.* 7000 or more) but no more than a few hundred patient samples [1]; in this case, it is *highly probable* that the training set will be separable while eliminating all the way down to a few hundred features. Accordingly, as also discussed in [1], when the data is initially separable, if the data is high-d (*e.g.* with thousands of features), “backward elimination methods” – such as all methods we are evaluating herein – may be able to eliminate all the way down to *relatively few* features (*e.g.* tens; or few hundred) without losing separability, whereas for low-to-intermediate dimensioned data separability may be initially achieved but lost *relatively sooner* *e.g.* when *half* of the original features – rather than *relatively few* – is still left to eliminate. By contrast, as a third category, *low-d data* (*e.g.* with 20-30 features) may often *start* (and remain) nonseparable. In this third case, not only is it not possible to evaluate approaches such as LO that require separability but also in our work herein we instead focus on high- or intermediate-dimensioned data since they are the two cases feature selection is most urgently needed; accordingly, herein we have broken up our experimental comparisons into the first two categories above.

High-Dimensional Datasets: For this category, defined above, we used the same datasets as previously used by [1], having thousands of features and only tens of samples (*i.e.* the gene datasets *Duke Breast Cancer*, *Leukemia*, *Colon Cancer*, obtained from the LIBSVM website [5] with 7129, 7129, 2000 features, respectively, and 38, 62, and 44 samples respectively) and a fourth dataset (*UCI Arcene*) with a somewhat high number of features (300 features). For the *very high-d* case (*i.e.* the three gene datasets), average test set classification error rate (*i.e.* an average across multiple trials) as a function of the number of retained features (which is reduced going from right to left) is shown in Figures 1, 4, and 3, *for two different nonlinear kernels* (Gaussian and polynomial)²². Notice from these six plots that the methods have formed four well-separated clusters *i.e.* tiers, as follows. The lowest two curves shown – MFE-LOemb and our proposed method BMFE-QPemb – form the best-performing tier (*i.e.* the smallest classification error rate). The next best tier – tier 2 – is formed by our other proposed method MFE-QPemb. Next, methods that perform stepwise full SVM retraining (*i.e.* methods with “FRsub” in their name) have clustered to precisely form tier 3. Lastly, ‘basic MFE’ and MFE-Slack – which are the only two methods that do not perform any type of classifier retraining – have formed tier 4 where we see the poorest performance.

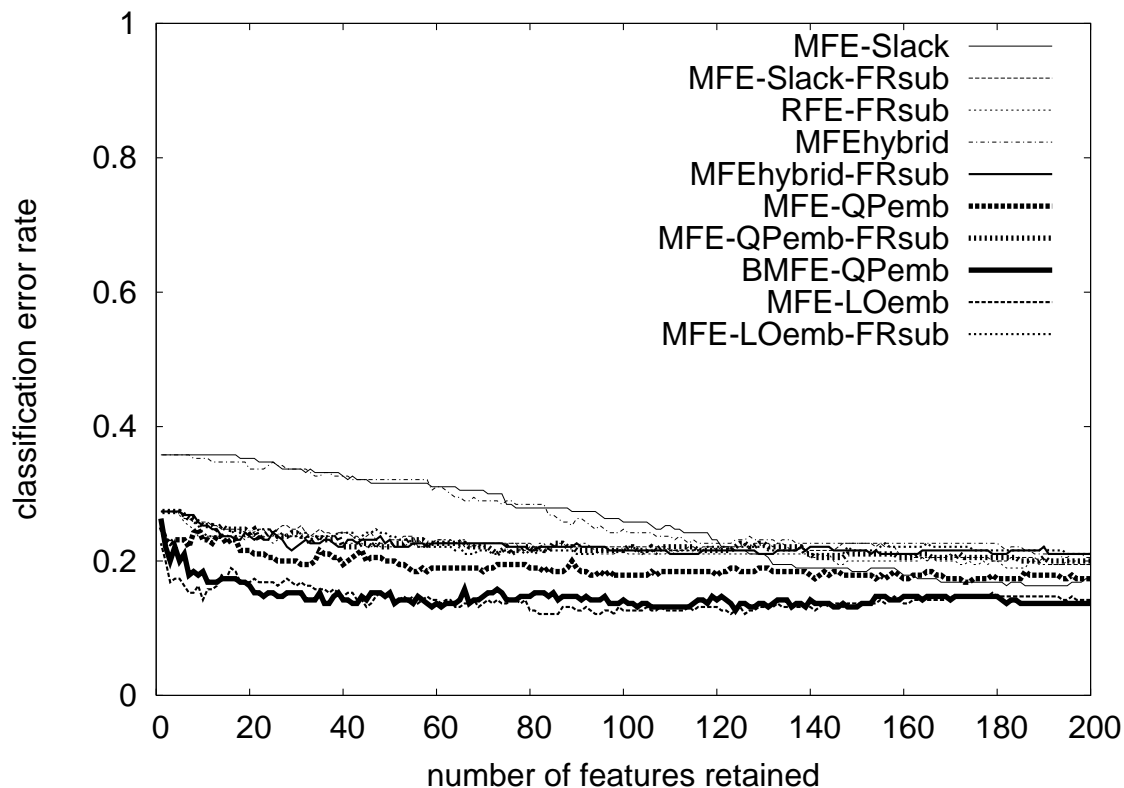
Several comments should be made at this point. First, as explained in an earlier Section above, the results we present herein for “full (SVM) retraining (FR)” and LO are more proper results than the results presented for these two *classifier retraining* approaches earlier in [1].²³ Keeping this in mind, notice that ‘basic MFE’ and MFE-Slack *i.e.* *non-retraining* methods, which had achieved the *best* performance in [1], are significantly outperformed here, forming the poorest tier – tier 4 – falling behind their FR-based variants (tier 3) as well as LO (tier 1). Second, let us now consider the two top performers – MFE-LOemb and our proposed method BMFE-QPemb – that are forming tier 1.

²²In these Figures (1, 4, 3), the number of trials averaged, respectively, are 5, 10, 10 for the Gaussian kernel and 4, 3, 8 for the polynomial kernel; they are the initially separable trials we generated.

²³Recall that LO is a type of (light) classifier retraining.

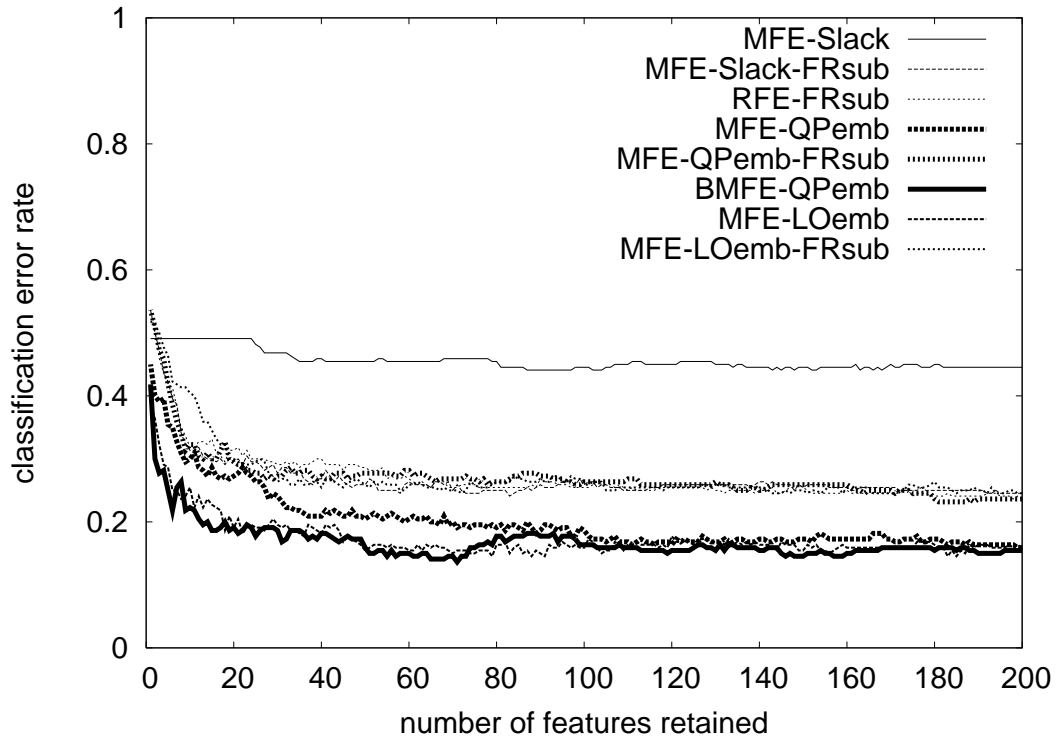


(a) Polynomial kernel.

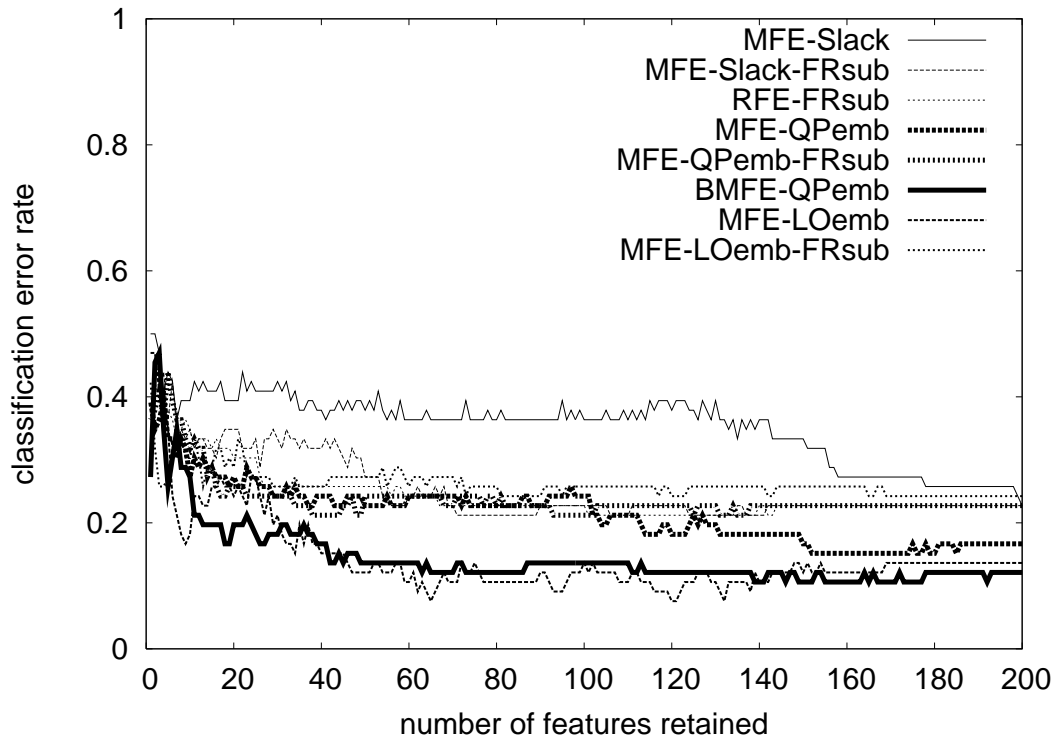


(b) Gaussian kernel.

Figure 3: Average test set classification error rate for the high-d *Leukemia* gene dataset that has 7129 initial features.

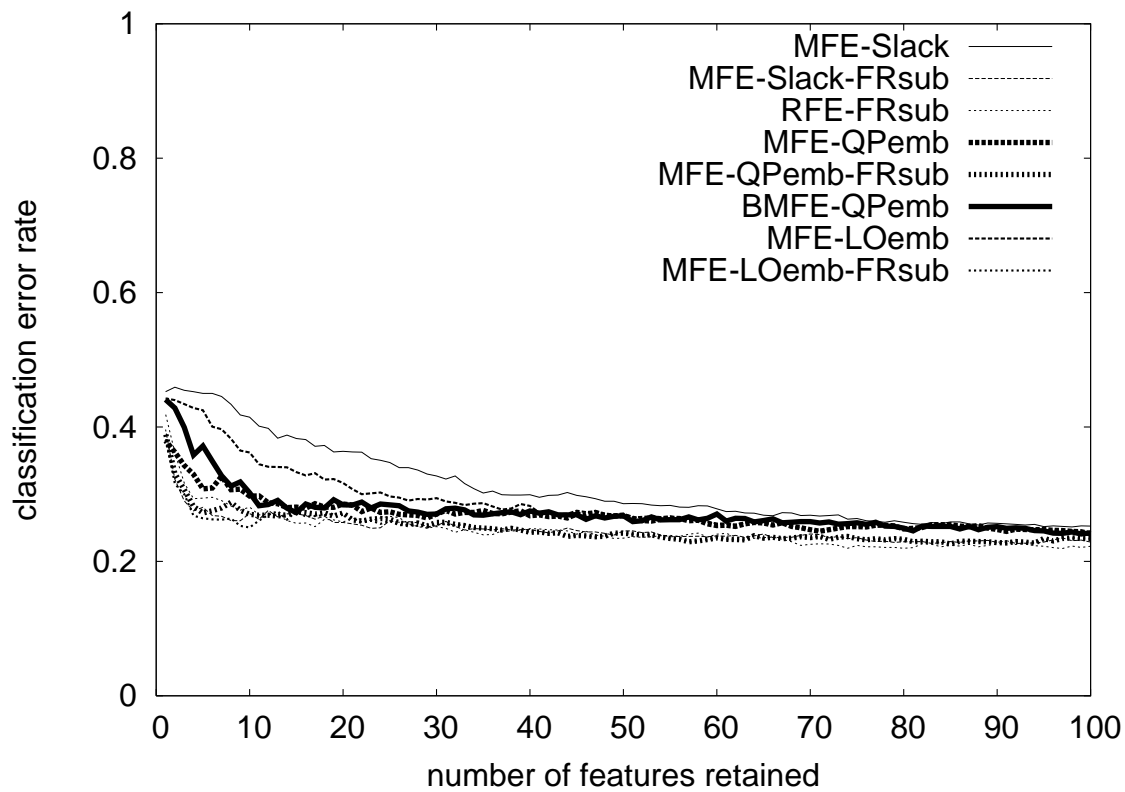


(a) Gaussian kernel.

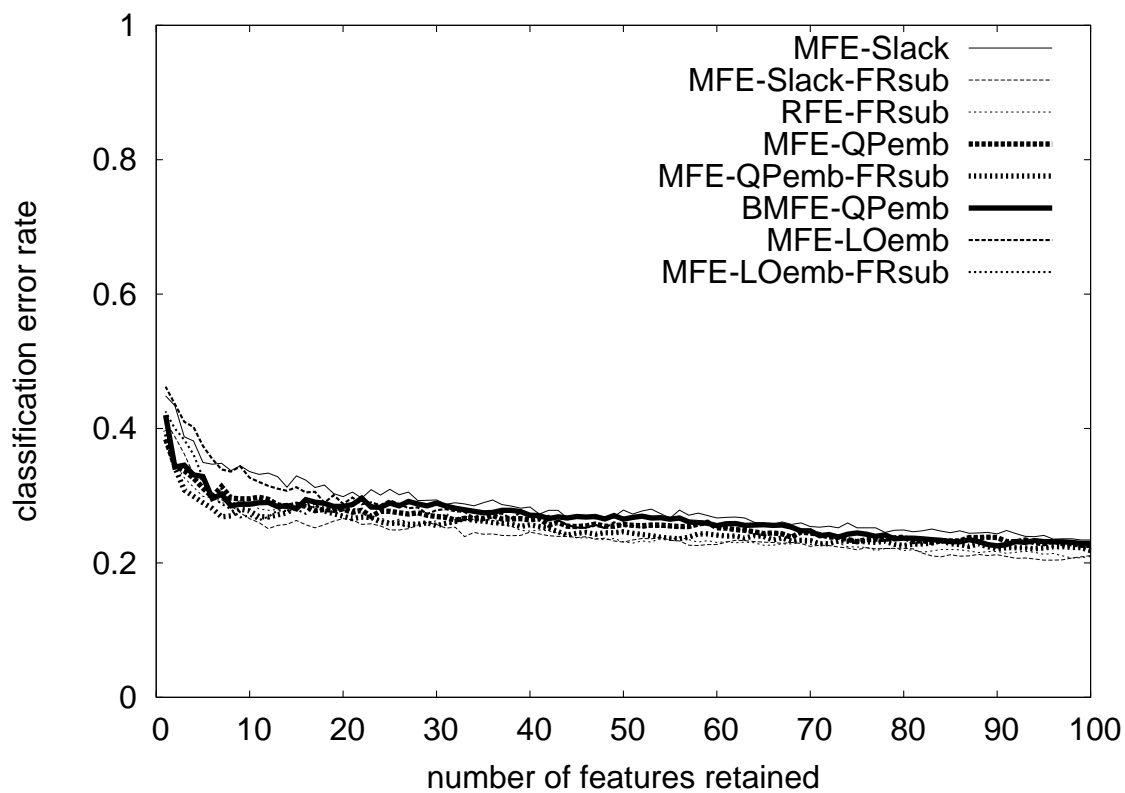


(b) Polynomial kernel with degree 3.

Figure 4: Average test set classification error rate for the high-d *Duke Breast Cancer* gene dataset that has 7129 initial features.



(a) Polynomial kernel.



(b) Gaussian kernel.

Figure 5: *UCI Arcene* dataset.

We begin by recalling that our experiments herein for method evaluations are actually somewhat favorably biased towards MFE-LOemb rather than our proposed BMFE-QPemb because, as we described earlier, while performing hyperparameter tuning during elimination may benefit BMFE-QPemb, herein we did not do this; in future work we may, in order to explore whether BMFE-QPemb improves upon such tuning, whereas, by contrast, MFE-LOemb performance is not going to change since such tuning is not applicable to the LO approach by definition. A second reason to advocate our proposed approach BMFE-QPemb is that separability – i.e. strictly satisfying the margin – which is *required* by MFE-LOemb (and not required by our proposed method BMFE-QPemb), may lead to overfitting when training samples at the margin are outliers or even *mis*labeled samples, as mentioned earlier. To evaluate this, in future work we may run experiments upon artificially mislabeling some samples. As a third supportive point for our proposed method BMFE-QPemb, note that it would be worthwhile to explore in future work a hybrid of these two best-performing methods; e.g. BMFE-QPemb can be used at elimination steps at which data separability has been lost, with MFE-LOemb used at the other (separable) steps.

Our final dataset for the high-d category, *UCI Arcene*, has a *somewhat high* number of features: only 300, not thousands, taken from the pool of available features for this dataset. Here the result, as shown in Fig. 5²⁴, is that tier 3 – i.e. the tier of methods that perform full SVM retraining – as a whole has shifted downward and become the best-performing tier, outperforming the LO and QP approaches that, by contrast, only employ *light classifier retraining*. This is not surprising since generally feature elimination can potentially benefit from (*aggressive*) *full SVM retraining*, as seen here, if the number of features is not *too high* such as 300 as seen here (since, as we mentioned earlier, overfitting, being a cumulative effect, depends on how high a number of elimination steps are accumulating.)

Low-to-Intermediate Dimensioned Datasets: Defined above, this experiment category is for datasets which start the feature elimination process separable but, unlike the high-d category, *do not maintain that separability all the way down to relatively few features*; e.g. datasets that lose separability after only half of their features are eliminated, such as the *Splice Scale* dataset whose results we show in Fig. 6²⁵. Notice in this Figure that our proposed method BMFE-QPemb remains a best performer whereas MFE-LOemb does not, because MFE-LOemb, unlike in the high-d case, is no longer *usable (applicable) for the vast majority of the elimination steps* (due to separability being lost after the elimination of half of the features).

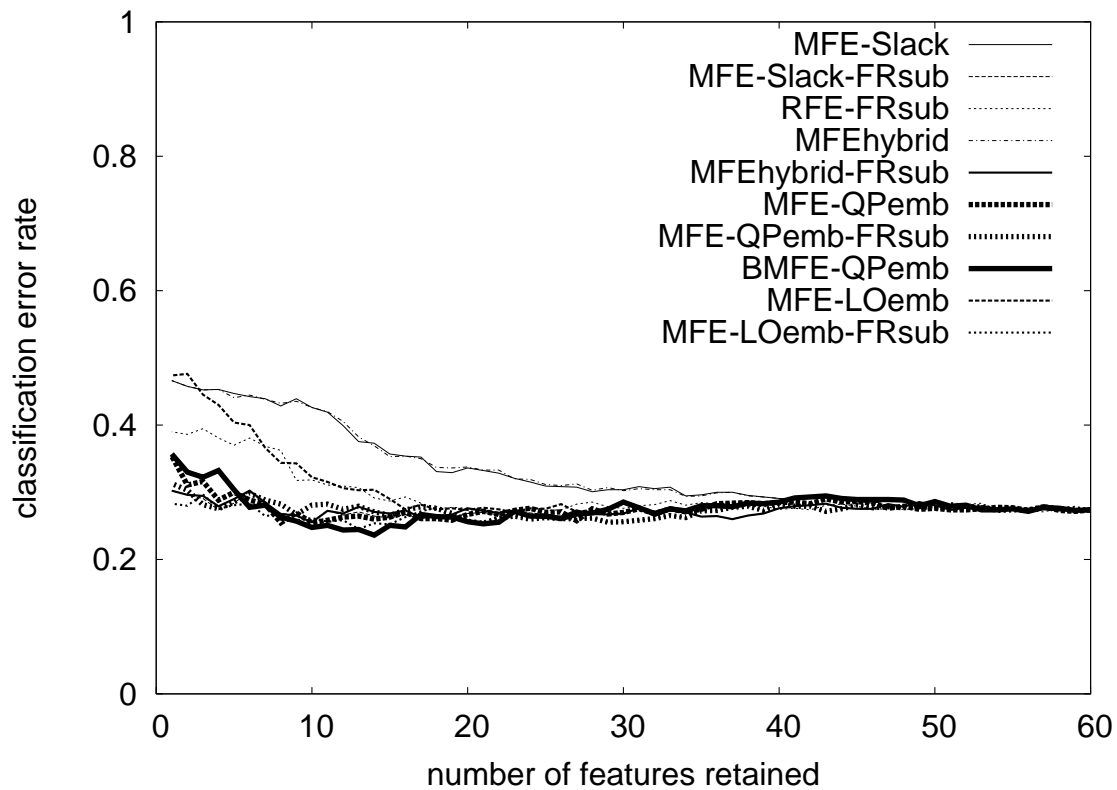
The full set of results we presented herein – for five different datasets and two different nonlinear kernels – show that our proposed methods achieve their best performance when data dimensionality is high, in the thousands. In fact, when data is high-d, notice that *our proposed method BMFE-QPemb is the best-performing method among all methods for the majority of experiments* (i.e. for 4 out of 6 distinct (dataset, kernel) pairings presented in Figures 4, 3, 1).

5 Conclusions

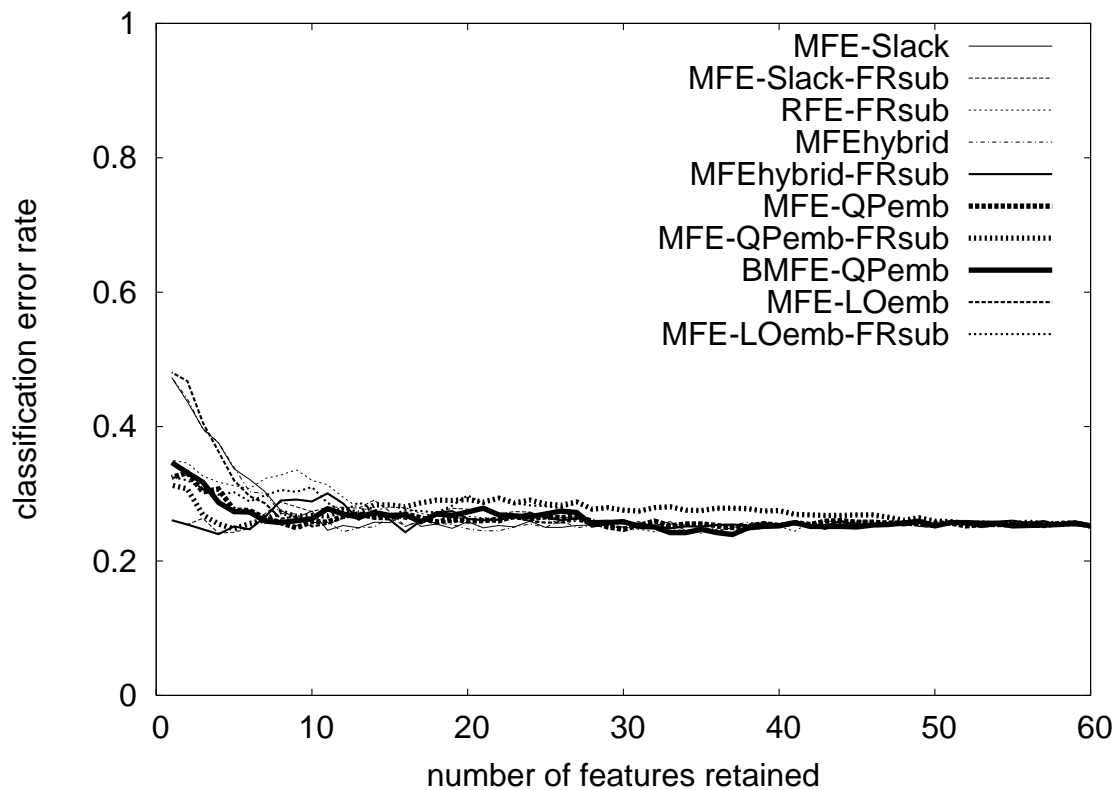
For SVM-based feature selection, we introduced two new, novel, fast and computationally low-cost, “backward feature elimination” approaches theoretically motivated by Statistical Learning Theory and margin maximization central to SVM learning, and showed that these approaches enable achieving higher or competitive generalization accuracy – lower or similar test set classification error rate – than previously proposed SVM-based MFE (Margin-Maximizing Feature Elimination) methods. Our first proposed approach, named “QP”, performs lightweight SVM retraining – equivalent to *training an SVM for 1-dimensional data* (which has very little computational cost) – that adjusts the current solution in the reduced feature space (*i.e.* upon candidate feature elimination) to aim to obtain a lower value for the SVM objective function in the reduced space. This approach

²⁴The number of trials averaged was 22 for the Gaussian kernel and 24 for the polynomial kernel – they are the initially separable trials among the 30 trials we generated.

²⁵The number of trials averaged was 14 for the Gaussian kernel and 13 for the polynomial kernel – they are the initially separable trials among the 30 trials we generated. We obtained the *Splice Scale* dataset from the LIBSVM [5] website.



(a) Polynomial kernel.



(b) Gaussian kernel.

Figure 6: Average test set classification error rate for the low-to-intermediate dimensioned *Splice Scale* dataset.

is consistent with theory; *i.e.*, the function we are minimizing – to slightly alter the decision boundary – is essentially the same one minimized by the standard SVM formulation. QP differs from the previously proposed margin-maximizing (i.e. MFE-based) approach LO (Little Optimization) due to being usable even when the data has become nonseparable during the feature elimination process; unlike LO, we treat *both* the SVM slackness parameters and the SVM hyperplane intercept as parameters to be optimized, conveniently via a very low-cost optimization formulation. QP also differs from another previously proposed MFE-based approach, MFE-Slack, due to allowing the decision boundary to be *altered* – lightly, not aggressively – during the feature elimination process. To embed this novel QP approach into the feature elimination decision – consistent with margin maximization – we proposed a feature elimination method named MFE-QPemb. We found that the generalization accuracy achieved by this method is higher than MFE-Slack’s but lower than that of the previously proposed LO-based MFE method (named MFE-LOemb herein).

Our second proposed approach BMFE (Bound-Based Margin-Maximizing Feature Elimination) is motivated by a well-known theoretical upper bound in Statistical Learning Theory – by Vapnik et al. – on the “expected risk” of making a classification error. Combining BMFE with the above approach of embedding QP into the elimination decision, while staying consistent with margin maximization central to SVM learning, we proposed the feature elimination method BMFE-QPemb, where the computation cost is, again, like our first method above, little, *i.e.* essentially again equivalent to 1d SVM training. We found that this BMFE-QPemb method brought an improvement in generalization accuracy over our MFE-QPemb method proposed above, consistently achieving generalization accuracy that is higher than or similar to MFE-LOemb’s, for *high-dimensional data, for especially which feature selection is urgently needed*.

In summary, for high-dimensional data, which is becoming more widespread in practice, we introduced a feature selection method named BMFE-QPemb – whose premise, consistent with margin maximization, is both *light retraining of the classifier* and consistency with a well-known bound on “expected risk” (of making a classification error) in Statistical Learning Theory – and showed that our proposed method achieves higher or competitive generalization accuracy – lower or similar test set error rate – than earlier MFE-based feature selection methods and RFE. We found this to be true even when these earlier methods stepwise employ the *computationally more costly* approach of *full SVM retraining* (*i.e.* stepwise recalculation of *all* SVM weights from scratch). In particular, we found that our proposed BMFE-QPemb method and the previously proposed MFE-LOemb method are the two best performers with respect to generalization accuracy, and noted some limitations of the previous method MFE-LOemb, such as *i)* not being usable whenever the data becomes nonseparable during the feature elimination process, *ii)* potential overfitting in the potential event the training samples at the margin are outliers or even *misabeled* samples, and *ii)* not incorporating a means of hyperparameter tuning into the elimination criterion.

6 Appendix: A fast algorithm for training a 1d SVM

Rather than solving the 1d SVM in the usual way an SVM is solved (by using a “dual” solver), an SVM training algorithm specifically designed for significantly reduced computation for the 1d case can be utilized. The algorithm²⁶, which is a simple one, is based on two key SVM properties unique to 1d data:

P1) In each class, there is *only one (and exactly one)* “margin-setter”, the training sample z_j located at margin distance to the boundary, *i.e.* the sample whose discriminant function value g_j (*i.e.* $y_j(wz_j + b)$) is equal to 1.²⁷ It can be easily seen from this equality that w is $2/r$ where r

²⁶Much of the algorithm has been previously documented in [19], a brief 4-page technical report document that we found online at the website of one of the authors.

²⁷To see why this holds, consider the following SVM details [3, 19]. In 1d, each support vector in class 1 has the same coordinate (and similar result holds for the other class) because in 1d there is only one scalar z_j that satisfies the above margin-setter equation $y_j(wz_j + b) = 1$. If we suppose there are *multiple* support vectors (which, by the above requirement, would be at the same coordinate), an identical solution can be produced by setting the Lagrange multiplier of one to a (positive) number and the rest to 0 [19].

is simply the (scalar) subtraction of one class’s margin-setter from the other’s; and subsequently b can be computed as $y_j - wz_j$ using *any sample z_j with zero slackness*. Thus, given a *candidate margin-setter pair* (where each such pair consists of one margin-setter per class), computing the associated w and b only takes two scalar additions and multiplications.

P2) The number of “margin-violators” (i.e. training samples with nonzero slackness) is *equal for the two classes*.²⁸

Accordingly, the algorithm to train the 1d SVM, with four steps, is:

Step 1) Consider these two distinct scenarios: S1) assume class 1 lies to the right of class 2; S2) vice versa. For each scenario, run Steps 2 and 3.

Step 2) Determine the scenario’s set of *candidate margin-setter pairs*, where each pair consists of one margin-setter sample per class; i.e., among all pairs of margin-setters (with one setter per class) that are suitable for P1, determine those that additionally meet P2.

Step 3) Specific to this particular scenario (S1 or S2), choose the candidate that minimizes the SVM objective function $w^2/2 + C \sum \xi_n$.

Step 4) Among the two winning candidates – one per scenario – determined above, choose as final the one that produced the smaller objective function value.

To minimize computation for the task of determining *all* candidate pairs for *Step 2*, one can first determine the one whose members (the two per-class margin-setters) lie *closest* to each other, because, given a pair a guaranteed next one is simply the next two samples – one sample per class – that are located in the *direction towards the opposite class*. Thus, one would first sort within-class samples in that direction, as a “pre-processing” step for the algorithm.²⁹ By nature of the specific sorting order, the two samples that make up a candidate pair – denoted here as sample p for class 1 and sample q for class 2 – share the same post-sorting (within-class) index, which thus also serves as pair index.³⁰ Given a pair index j , the set of margin-violators, which is needed by the sum in *Step 3*, is *readily* identified by indexes $k > j$. This sum can thus be written as $\sum_n \xi_n = \sum_{k>j} (1 - (p_k w + b)) + \sum_{k>j} (1 + (q_k w + b)) = 2(N - j) - w(d_j^+ - d_j^-)$, where $d_j^+ \equiv \sum_{k>j} p_k$ and $d_j^- \equiv \sum_{k>j} q_k$. Since $d_j^+ = d_{j+1}^+ + p_{j+1}$ and $d_j^- = d_{j+1}^- + q_{j+1}$, which are recursive equations, the sum $\sum_n \xi_n$ of *Step 3* can be computed *recursively* (to reduce computation).

References

- [1] Y. Aksu, D. J. Miller, G. Kesidis, and Q. X. Yang, “Margin-Maximizing Feature Elimination Methods for Linear and Nonlinear Kernel-Based Discriminant Functions”, *IEEE Trans. Neural Netw.*, vol. 25, no.10, pp.701-717, 2010.

²⁸To see why P2 holds, consider the following SVM details [3, 19]. Each sample i has two, nonnegative Lagrange multipliers: μ_i for constraint $\xi_i \geq 0$, and λ_i for constraint $y_i(wz_i + b) \geq 1 - \xi_i$. By KKT conditions $\lambda_i + \mu_i = C$ and $\mu_i \xi_i = 0$, note that *i)* $\lambda_i = C$ for any sample with nonzero slackness (a “margin-violator”) and *ii)* $0 < \lambda_i \leq C$ for a support vector (margin-setter; this result is due also to P1). For class 1 and 2, respectively, let λ^+ and λ^- denote the multipliers for support vectors, and let N^+ and N^- denote the number of margin-violators (whose λ is C). By KKT condition $\sum_j \lambda_j y_j = 0$ and P1, note that $\lambda^+ - \lambda^- = C(N^- - N^+)$.

This quantity (r.h.s.) being 0 (i.e., P2 not holding) would imply $|\lambda^+ - \lambda^-| \geq C$, which would imply (due to $\lambda_i \leq C$ in *ii* above) that either λ^+ or λ^- would have to be 0 which contradicts *ii* above [19]. Thus, P2 holds.

²⁹Thus, for S1, the sorting is in descending order for class 1, and in ascending order for class 2; and the sorting orders for S1 and S2 are the opposites of each other.

³⁰Note that seeking the *initial* pair (i.e. the pair of “*closest*” samples) only involves *comparison of one scalar to another* i.e. essentially no computation. To qualify as a valid margin-setter candidate for class 1, a sample needs to be smaller than (lie to the *left* of) its counterpart in class 2 (*if S1*; for S2, the inverse logic is required). This test starts with the two samples at extreme ends of their classes and successively proceeds to consecutive pairs until a pair is found that passes the test.

- [2] Y. Aksu, D. J. Miller, G. Kesidis, D. C. Bigler, and Q. X. Yang, "An MRI-Derived Definition of MCI-to-AD Conversion for Long-Term, Automatic Prognosis of MCI Patients", *PLoS ONE*, 6(10):e25074. doi:10.1371/journal.pone.0025074, 2011.
- [3] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining Knowledge Discovery* 2 (2) pp.121-167, 1998.
- [4] A. B. Chan, N. Vasconcelos, and G. R. Lanckriet, "Direct convex relaxations of sparse SVM," in *Proc. Int. Conf. Machine Learning*, 2007.
- [5] C. Chang and C. Lin, "LIBSVM : a library for support vector machines," software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2001.
- [6] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, June 1965.
- [7] C. Davatzikos, P. Bhatt, L. M. Shaw, K. N. Batmanghelich, J. Q. Trojanowski, "Prediction of MCI to AD conversion, via MRI, CSF biomarkers, and pattern classification", *Neurobiology of Aging*, 2010, doi:10.1016/j.neurobiolaging.2010.05.023.
- [8] R. Duda, P. Hart, and G. Stork, Pattern Classification. Second Edition, John Wiley and Sons, New York, 2001.
- [9] G. Fung, O. L. Mangasarian, "A feature selection Newton method for support vector machine classification," *Computational Optimization and Applications*, vol. 28, no.2:185-202, July 2004.
- [10] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157-1182, 2003.
- [11] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, 46(1):389-422, 2002.
- [12] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, New York: Springer, 2001.
- [13] Computational Methods of Feature Selection, edited by H. Liu and H. Motoda, Chapman & Hall/CRC, 2008.
- [14] O. L. Mangasarian, "Exact 1-norm support vector machines via unconstrained convex differentiable minimization," *J. Mach. Learn. Res.*, vol. 7, pp. 1517-1530, 2006.
- [15] C. Misra, Y. Fan, C. Davatzikos, "Baseline and longitudinal patterns of brain atrophy in MCI patients, and their use in prediction of short-term conversion to AD: Results from ADNI", *NeuroImage* 44, pp.1415-1422, 2009.
- [16] D. Mladenić, J. Brank, M. Grobelnik, N. Milic-Frayling, "Feature selection using linear classifier weights: interaction with classification models," *Proc. ACM SIGIR Conf. on R&D in Info. Retrieval*, pp. 234-241, 2004.
- [17] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. PAMI*, vol. 27, no. 8, pp. 1226-1238, Aug. 2005.
- [18] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latiluppe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T. R. Golub, "Multiclass cancer diagnosis using tumor gene expression signatures," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 26, 15149-15154, Dec. 2001.

- [19] Y. Su, T. M. Murali, V. Pavlovic, and S. Kasif, "Training Support Vector Machines in 1D", December 8, 2002; a technical report obtained from author website.
- [20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Statistical Society, Series B (Methodological)*, vol. 58, no. 1, pp. 267-288, 1996.
- [21] G. V. Trunk, "A problem of dimensionality: A simple example," *IEEE Trans. PAMI*, vol. 1, issue 3, pp. 306-307, July 1979.
- [22] V. Vapnik, The Nature of Statistical Learning Theory. Springer, New York, 1996.
- [23] V. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
- [24] Y. Wang, Y. Fan, P. Bhatt, C. Davatzikos, "High-dimensional pattern regression using machine learning: From medical images to continuous clinical variables," *NeuroImage* 50, pp.1519-1535, 2010.
- [25] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, "Feature selection for SVMs," *NIPS 13*, MIT Press, 2001.
- [26] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-norm support vector machines," *NIPS*, 2003.